Section 9
Tool Manager

The *ETMS User Functions* can be accessed by means of the *Tool Manager*. The *ETMS User Functions* provide the interface between the *ETMS* and the user. These functions include the *Aircraft Situation Display (ASD)*, *EMail*, *Delay Manager*, and *TM Shell*.

9. Tool Manager

Tool Manager (*TMGR*) is a graphical user interface (GUI) program that helps a traffic management specialist use ETMS software. *TMGR* is used for quick access to ETMS software without interacting with the complexity of the operating system. With the use of graphical symbols, the traffic management specialist can start any ETMS software. *TMGR* is an interactive tool and responds to traffic management specialist requests as entered through the use of the keyboard and mouse/trackball. At least one *TMGR* runs on each traffic management specialist workstation.

NOTE: It is recommended that the reader become thoroughly familiar with the operation of the *TMGR* before reading this section. The operation of *TMGR* is described in Chapter Four of the *System Administration Manual*.

The *Tool Manager* function consists of a single process.

Design Issue: Graphics Support Software

TMGR uses the Apollo *Graphics Primitives (GPR)* package to manage all graphics, keyboard input, and mouse input. The *GPR* package is the lowest-level graphics support software provided by Apollo. *GPR* was chosen because it allows for customizing the use of the graphics hardware for optimal performance. *GPR* is fully integrated with the Apollo operating system and display manager software.

GPR is used in *direct* mode, that is, the program operates within a display manager window; it does not borrow the display. The main reasons for using direct mode is to allow other Apollo shells to co-exist with *TMGR*.

The processing of *TMGR* is heavily dependent on the *GPR* routines. The reader should refer to the Apollo *GPR* documentation as needed.

Execution Control

The execution of *TMGR* is initiated from a script or the function key **F0**. The script may be a manually invoked shell script or a *login* script. The quick key **F0** is a script that defines the function key to start *TMGR* if *TMGR* is not running. The quick key **F0** "pops" *TMGR* to the top of the display if it is an icon or is covered by other ETMS software.

Input

TMGR receives dynamic mail message data from other ETMS functions. *TMGR* processes the following mail messages:

- Statistical requests
- Reconfigure commands
- Start commands

Start commands allow a remote user to start any of the icon records and menu items controlled by *TMGR*.

TMGR operates by values assigned and defined in the **setup** file. The default **setup** file is named: /tmgr/config/tmgr.default. The **setup** file describes what is displayed on *TMGR* and what to execute when icons and menu items are selected. Variables allow each *TMGR* executed to be configured for different needs or appearances. All variables and values can be upper or lower case. To assign a value to a variable, the variable must be followed by **:=** and then a value; for example, **variable := value** .

Comments are placed in the **setup** file by placing the symbol **#** as the first letter on the line. Comments make the file readable for inexperienced users.

Setup File Variables. *TMGR* reads any of the following variables from the **setup** file when it is initially invoked:

animate_icons - A TRUE or FALSE value. TRUE causes icons to be animated when the cursor is on them. The icon will *animate* between two or more icon representations. FALSE prevents animation. The default value is FALSE.

animate_speed - A decimal number value. The number specifies how long, in seconds, between animation drawings while the cursor is on an icon. The default value is two seconds. This variable is dependent on a value of TRUE for **animate_icons**.

children_hot_key - An upper or lower case letter, number, or the word **control_*n***, where *n* represents a letter. This variable specifies the hot key to toggle on or off the display of the children icons. The default value is **control_k**.

children_icons - A TRUE or FALSE value. TRUE causes *TMGR* to display a child icon for each version of a process that is invoked. The default value is TRUE.

digital_clock_on - An **ON** or **OFF** value. The value specifies whether a digital clock is displayed in the title bar area. The default value is TRUE.

digital_clock_position - A **RIGHT** or **LEFT** value. The value specifies where the digital clock is positioned in the title bar area. The default value is the **RIGHT** hand side of the title bar area.

digital_clock_time - A **UTC**, **LOCAL**, or **EST** value. The value specifies whether the digital clock displays Coordinated Universal Time (UTC), local or Eastern Standard Time (EST). The default value is **UTC**.

digital_clock_24hour - A TRUE or FALSE value. The value specifies whether the digital clock displays time in 24-hour or 12-hour format. TRUE sets the 24-hour format. The default value is TRUE (24-hour format).

digital_date - A TRUE or FALSE value. The value specifies whether the date is displayed with the digital clock in the title bar area. If the **digital_clock_on** variable is set to TRUE, the date is displayed to the left of the clock when it is displayed on the right . It is displayed to the right when the clock is on the left. The default value is TRUE.

function_keys_on - A TRUE or FALSE value. TRUE allows the user to use function keys **F0** through **F9** to invoke icons. **F0** invokes the prompt, while **F1** through **F9** invoke the icons shown, beginning with the left, topmost icon. FALSE causes the function keys to act as they usually do. The default value is TRUE.

help_file_region - Four decimal numbers separated by a space. The numbers define the location of the corner coordinates of the help file window. The first two numbers specify the **x** and **y** coordinates for the upper left corner of the window. The next two numbers specify the **x** and **y** coordinates for the lower right corner of the window. The default value is **0**, **0**, **500**, **500**.

help_icon_char - An upper or lower case letter or number. A maximum of 10 letters or numbers is used (separated by a space). This variable specifies icon character(s) from the library font to display the *help* icon. The multiple values of the variable allow the animation of the icon. The default value is the letter **Q**, which causes the help icon to be displayed as a question mark (?).

help_icon_on - A TRUE or FALSE value. If the value is TRUE, *TMGR* displays the *help* icon. Selecting the icon displays a help file. The default value is TRUE.

icon_bg_color - One of the following colors are accepted : **black, red, green, blue, cyan, yellow, magenta, white, pale green, dark green, reddish tan, brick red, pale blue, dull blue, khaki** or **brown**. This variable specifies the background color within *TMGR* icons. The default value is **khaki**.

icon_spacing - A decimal number value between one and ten. This value specifies the number of pixels between icons. The default value is 10 pixels.

icon_text_color - One of the following colors is accepted : **black, red, green, blue, cyan, yellow, magenta, white, pale green, dark green, reddish tan, brick red, pale blue, dull blue, khaki** or **brown**. This variable specifies the color of text or graphics used in *TMGR* icons. The default value is **black**.

invoke_icon_char - An upper or lower case letter or number. A maximum of 10 letters or numbers is used (separated by a space). This variable specifies icon character(s) from the library font to display the *invoke* icon. The multiple values of the variable allow the animation of the icon. For example, ??? The default value is the symbol asterisk (*).

invoke_icon_on - A TRUE or FALSE value. TRUE causes *TMGR* to display the *invoke* icon. Selecting the icon displays a prompt that allows users to invoke Shell commands. The default value is FALSE.

invoke_region - Four decimal numbers separated by a space. The numbers define the location of the corner coordinates of the invoke window. The first two numbers specify the **x** and **y** coordinates for the upper left corner of the window. The next two numbers specify the **x** and **y** coordinates for the lower right corner of the window. The default value is **0**, **0**, **500**, **500**.

lock_position - A TRUE or FALSE value. TRUE causes *TMGR* to reposition to the original position when the tool was executed whenever the *TMGR* window is moved. If the value is FALSE, the *TMGR* window can be moved to any location on the display. The default value is FALSE.

menu_bg_color - One of the following colors are accepted: **black, red, green, blue, cyan, yellow, magenta, white, pale green, dark green, reddish tan, brick red, pale blue, dull blue, khaki**, or **brown**. This variable specifies the background color for individual items on menus displayed by *TMGR*. The default value is **khaki**.

menu_font_title_path - The name of a font file, *x* characters long. This file specifies the location of the font library used for the titles of menus displayed by *TMGR*. This library contains regular alphanumeric characters. The name of the file can be a complete pathname or by itself. If the file is not a complete pathname, the location of the font file is assumed to be in the /tmgr/fonts directory. The default file for this variable is **/tmgr/fonts/tmgr_menu.font**.

menu_text_color - One of the following colors are accepted: **black, red, green, blue, cyan, yellow, magenta, white, pale green, dark green, reddish tan, brick red, pale blue, dull blue, khaki**, or **brown**. This variable specifies the text color for individual items on menus displayed by *TMGR*. The default value is **black**.

menu_title_bg_color - One of the following colors are accepted: **black, red, green, blue, cyan, yellow, magenta, white, pale green, dark green, reddish tan, brick red, pale blue, dull blue, khaki**, or **brown**. This variable specifies the background color for title area of menus displayed by *TMGR*. The default value is **dull blue**.

menu_title_text_color - One of the following colors are accepted: **black, red, green, blue, cyan, yellow, magenta, white, pale green, dark green, reddish tan, brick red, pale blue, dull blue, khaki**, or **brown**. This variable specifies the text color for the title of menus displayed by *TMGR*. The default value is **khaki**.

orientation - The variable is assigned the words **horizontal** or **vertical** or the letters **h** or **v.** This variable specifies which way the row of icons are displayed, **horizontal** or **vertical**. The default value is the **horizontal** orientation.

quit_icon_char - An upper or lower case letter or number. A maximum of 10 letters or numbers is used (separated by a space). This variable specifies icon character(s) from the library font to display the *quit* icon. The multiple values of the variable allow the animation of the icon. For example, ??? The default value is the letter **E**.

quit_icon_on - A TRUE or FALSE value. TRUE causes *TMGR* to display the *quit* icon. Selecting the icon exits *TMGR*. The default value is TRUE.

read_setup_hot_key - An upper or lower case letter, number, or the word **control_n**, where **n** represents a letter. This variable specifies the hot key to re-read the *TMGR* **setup** file and put any changes into effect. The default value is **control_r**.

title - Words, letters, or numbers to display at the top of the *TMGR* window, with a length of *x* characters. There is no default value for this variable.

title_bg_color - One of the following colors is accepted: **black, red, green, blue, cyan, yellow, magenta, white, pale green, dark green, reddish tan, brick red, pale *b*lue, dull blue, khaki**, or **brown**. This variable specifies the background color for the *TMGR* title area. The default value is **khaki**.

title_text_color - One of the following colors is accepted: **black, red, green, blue, cyan, yellow, magenta, white, pale green, dark green, reddish tan, brick red, pale blue, dull blue, khaki**, or **brown**. This variable specifies the text color for the *TMGR* title. The default value is **black**.

tmgr_bg_color - One of the following colors is accepted: **black, red, green, blue, cyan, yellow, magenta, white, pale green, dark green, reddish tan, brick red, pale blue, dull blue, khaki**, or **brown**. This variable specifies the background color of the main *TMGR* window. The default value is **dull blue**.

tmgr_icon_path - The name of a font file, *x* characters long. This file specifies the location of the library where the icon character(s) that are used for the *invoke*, *help*, and *quit* icons. The name of the file can be a complete pathname or by itself. If the file is not a complete pathname, the location of the font file is assumed to be in the /tmgr/fonts directory. The default file for this variable is **/tmgr/fonts/tmgr_icon.font**.

Icon Record Variables. The variables on the previous pages set *TMGR*'s colors and also the Help, Invoke, or Quit icons To further customize *TMGR*, an icon record is defined. Defining icon records allows users to start one or more different ETMS programs.

To begin a record, the line should contain the **icon_record:** variable. All variables beneath this line are associated with this one icon record. A maximum of 30 icons can be defined.

A special icon record named **default_icon_record:** can be declared to declare variables once rather than having copies throughout the **setup** file. This record uses the same variables as the icon record (except menu items). If a variable in an icon record is not declared but declared in the default, the icon record will use the value assigned in the default icon record. The end of an icon record is found when another icon record or default icon record is declared. Comments, the assignment symbol (**:=**), and upper and lower case rules still apply.

TMGR reads any of the following variables for icon records:

auto_close - A TRUE or FALSE value. TRUE causes *TMGR* to automatically close and make the window disappear when the invoked process has exited. The default value is TRUE.

auto_quit - A TRUE or FALSE value. TRUE causes *TMGR* to quit when this icon record is selected. The default value is FALSE.

help_file - The name of an ASCII file. This file is displayed for help information when the icon is selected with the right mouse/trackball button. The name of the file can be a complete pathname or by itself. If the file is not a complete pathname, the location of the help file is assumed to be in the **/tmgr/data/help** directory. The default file for this variable is **/tmgr/data/help/tmgr_help**.

hot_key - An upper or lower case letter, number or the word *control_n*, where *n* represents a letter. This variable specifies the key used to invoke the data in the icon without selecting the icon with the mouse/trackball. There is no default value associated with this variable.

icon_char - An upper or lower case letter or number. A maximum of 10 letters or numbers is used (separated by a space). This variable specifies icon character(s) from the icon library that is used to display this icon. The multiple values of the variable allow the animation of the icon. For example, ??? There is no default value associated with this variable.

icon_path - The name of a font file, *x* characters long. This file specifies the location of the library where the icon character(s) are that are used to display the icon. The name of the file can be a complete pathname or by itself. If the file is not a complete pathname, the location of the font file is assumed to be in the **/tmgr/fonts** directory. The default file for this variable is **/tmgr/fonts/tmgr_icon.font**.

icon_word - A maximum of three words separated by the vertical bar symbol (|). The words will display on the icon rather than the graphic symbol declared in the variable **icon_char**. There is no default value associated with this variable.

menu_title - A message that can contain up to 256 letters or numbers. The information declared in this variable will be displayed at the top of all menu items when the icon is selected. There is no default value associated with this variable.

no_window - A TRUE or FALSE value. TRUE for this icon record does not create a window for invoking a specified process. This variable is used to provide a window for doing Display Manager commands or it is used by ETMS software that creates its own windows in which to display execution progress. The default value for this variable is FALSE.

number_process_children - A number value. This variable defines the maximum number of processes that this icon record can start. The maximum value for this variable is **6**. The default value for this variable is **6**.

process_invoke - The name of the program to execute, *x* characters long. This variable defines the action that the system is to perform whenever this icon is selected. It usually begins with **/com/sh** to create a process shell. It is then followed by the command, pathname of script, or name of ETMS software to be invoked in that shell. There is no default value associated with this variable.

process_name - A word that describes what this icon record starts. For example, if this icon record starts the ETMS program NET.MAIL, an appropriate name could be **mail** or **net_mail**. This variable is important for determining whether a process has exited or for *popping* the process to the front of the display. All icon records should have different values for this variable. There is no default value associated with this variable.

process_region - Four decimal numbers separated by a space. The numbers define the location of the corner coordinates of the process window. The first two numbers specify the **x** and **y** coordinates for the upper left corner of the window. The next two numbers specify the **x** and **y** coordinates for the lower right corner of the window. There is a maximum of six process regions that can be defined. Each region is separated by a semicolon. This variable is associated with the **number_process_children** variable so that the first process started corresponds to the first region declared. For each region not defined, the last region that is defined corresponds to the last process child that can be started. The default value is **0**, **0**, **5**, **5**.

send_message - A message that can contain up to 256 letters or numbers. This variable is used to send messages through the communication software running on the computer. This is similar to the **process_invoke** variable except that no ETMS software is being started but messages or codes are sent to other ETMS software. Therefore, either **send_message** or **process_invoke** can be defined for one icon record. There is no default value associated with this variable.

send_message_address - Decimal numbers or a correct alphanumeric word, *x* characters long, that designates an address with the communication software. To send messages between processes, each process must have a unique number or word. The format **10.29.30.02.0** is an example for addressing a process using decimal numbers. The numbers correspond to site, node, class, invocation, and sub address names, respectively.

The following words are used for describing a process address: **this**, **any**, **all** , **$site**, **$class**, **//node**. This is similar to the **process_invoke** variable except that no ETMS software is being started but messages or codes are sent to other ETMS software. Therefore, either **send_message_address** or **process_invoke** can be defined for one icon record. There is no default value associated with this variable.

send_message_code - A number value. This variable is used to send message codes (in conjunction with ASCII messages, as in the **send_message** variable) through the communication software running on the computer. This is similar to the **process_invoke** variable except that no ETMS software is being started but messages or codes are sent to other ETMS software. Therefore, either **send_message_code** or **process_invoke** can be defined for one icon record. There is no default value associated with this variable.

user_access - A user account or node name, *x* characters long. If this variable is defined, the icon record will work only for a particular user logged on the computer or a particular node name. The values for the variable are **user.account** or **//node**. There is no default value associated with this variable.

Menu Item Variables for Icons. There can be more than one **process_invoke** variable to each icon record by declaring menu items. Menu items allow more flexibility in starting the same process with different arguments. Menu items are defined inside an icon record. There is a maximum of 30 menu items allowed in an icon record.

To begin a menu item, the line should contain the **menu_item:** declaration. All variables following this declaration are associated with this one menu item. The menu item ends when another menu item, icon record, or default icon record is declared. A menu item can *inherit* values from the icon record or default icon record. The variables described on the following pages can inherit values. Comments, the assignment symbol (**:=**), and upper and lower case rules still apply.

TMGR reads any of the following variables for menu items:

auto_close - A TRUE or FALSE value. TRUE causes *TMGR* to automatically close and make the window disappear when the process invoked has exited. The default value is TRUE.

auto_quit - A TRUE or FALSE value. TRUE causes *TMGR* to quit when this menu item is selected. The default value is FALSE.

heading - A message that can contain up to 256 letters or numbers. The information in this variable will be displayed on the menu item when the icon is selected. There is no default value associated with this variable.

help_file - The name of an ASCII file, *x* characters long. This file is displayed for help information when the menu item is selected with the right mouse/trackball button. The name of the file can be a complete pathname or by itself. If the file is not a complete pathname, the location of the help file is assumed to be in the **/tmgr/data/help** directory. The default file for this variable is **/tmgr/data/help/tmgr_help**.

hot_key - An upper or lower case letter, number or the word **control_n**, where **n** represents a letter. This variable specifies the key used to invoke the data in the menu item without selecting the item with the mouse/trackball. There is no default value associated with this variable.

no_window - A TRUE or FALSE value. If the value is TRUE, this menu item does not create a window for invoking the specified process. This is used in doing Display Manager commands or ETMS software that creates its own windows to execute in. The default value for this variable is FALSE.

process_invoke - The name of the program to execute, *x* characters long. This variable defines the action that the system is to perform whenever this menu item is selected. Usually begins with **/com/sh** to create a process shell. It is then followed by the command, pathname of script, or name of ETMS software to be invoked in that shell. There is no default value associated with this variable.

process_region - Four decimal numbers separated by a space. The numbers define the location of the corner coordinates of the process window. The first two numbers specify the **x** and **y** coordinates for the upper left corner of the window. The next two numbers specify the **x** and **y** coordinates for the lower right corner of the window. There is a maximum of six process regions that can be defined. Each region is separated by a semicolon.

This variable is associated with the **number_process_children** variable in the icon record so that the first process started corresponds to the first region declared. For each region not defined, the last region that is defined corresponds to the last process child that can be started. The default value is **0**, **0**, **5**, **5**.

send_message - A message that can contain up to 256 letters or numbers. This variable is used to send messages through the communication software running on the computer. This is similar to the **process_invoke** variable except that no ETMS software is being started but messages or codes are sent to other ETMS software. Therefore, either **send_message** or **process_invoke** can be defined for one menu item. There is no default value associated with this variable.

send_message_address - Decimal numbers or a correct alphanumeric word, *x* characters long that designates an address with the communication software. To send messages between processes, each process must have a unique number or word. The format **10.29.30.02.0** is an example for addressing a process using the decimal numbers. The numbers correspond to **site**, **node**, **class**, **invocation**, and **sub address names**. The following words are used for describing a process address: **this**, **any**, **all** , **$site**, **$class**, **//node**. This is similar to the **process_invoke** variable except that no ETMS software is being started but messages or codes are sent to other ETMS software. Therefore, either **send_message_address** or **process_invoke** can be defined for one menu item. There is no default value associated with this variable.

send_message_code - A decimal number between 0 and 65,536. This variable is used to send message codes (in conjunction with ASCII messages, for example the **send_message** variable) through the communication software running on the computer. This is similar to the **process_invoke** variable except that no ETMS software is being started but messages or codes are sent to other ETMS software. Therefore, either **send_message_code** or **process_invoke** can be defined for one menu item. There is no default value associated with this variable.

Setup File Include Statement. To make the **setup** file more readable and modular, the **%include** statement is used. The parameter after the statement is another **setup** file that should be read. Inside this **include** file can be icon record(s), menu item(s), default icon record, or variables. The name of the file can be a complete pathname or by itself. If the file is not a complete pathname, the location of the file is assumed to be in the **/tmgr/config** directory.

TMGR reads the following optional arguments from the command line that initiates the program execution:

-h - This option displays all argument options.

-p { (**X**-coordinate, **left**, **right**, **center**) (**Y**-coordinate, **top**, **bottom**, **center**) } **-** This option positions the Tool Manager window when the program starts.

-path - This option retrieves and stores data at a different directory.

-s { name of **s**etup file } **-** This option uses a different tool manager **s**etup file.

-v4 - This option prevents connection to a node switch at startup time.

TMGR reads data from the following static data files:

tmgr_icon.font - This option contains the font used for the *help, invoke*, and *quit* icons.

tmgr_input.font - This option contains the font used for the invoke prompt.

tmgr_menu.font - This option contains the font used for the menu items.

TMGR can also use any user-specified or system font for displaying icons or menu items. This is done by assigning a value to the following variables: **icon_path**, **tmgr_icon_path**, or **menu_font_title_path**.

TMGR optionally reads data by assigning a value to the **help_file** variable, which is used in the default icon record, icon record, and menu item.

TMGR optionally reads data from the following file in response to user commands:

Setup file **-** This file contains the variables, icon records, and menu items read during startup.

Output

TMGR sends dynamic mail message data to other ETMS functions or to any ETMS software described by the **send_message_address** variable. The contents of the messages are described by the following variables:

send_message

send_message_code

If all these variables are declared and properly assigned, *TMGR* passes the information to the communication software running on the computer.

TMGR optionally writes data to the following files in response to user commands:

- A temporary help file **-** *TMGR* creates a file in the **/tmp** directory when the user requests a help file. This file is displayed in a window until *TMGR* quits or the user closes the window with the correct Display Manager commands. The file is deleted when the window is closed.

- A crash file - *TMGR* creates an error log file named **/tmgr/trace/tmgr_crash**. If the file already exists, the size of the file will be checked. If the size of the file is greater than 4K, the file will be deleted and a new file will be created. Whenever *TMGR* crashes because of an error or the user forces the program to crash (by pressing **Ctrl Q**), the program will write the time and date when the crash occurred and the current code that it was processing.

Processing Overview

Figure 9-1 shows that the processing performed by *TMGR* can be grouped into five main modules. The *Initialize* module starts by setting the cleanup handler, finding the data directory for retrieving and storing data, setting the default colors to the appropriate color variables, reading the startup arguments, connecting to the node switch, and initializing the graphics. The *Initialize* module displays a message to confirm if connected to a node switch. *TMGR* is now ready to read the variables that tell it what to display and perform. The *Configure* module performs the function of reading the **setup** file.

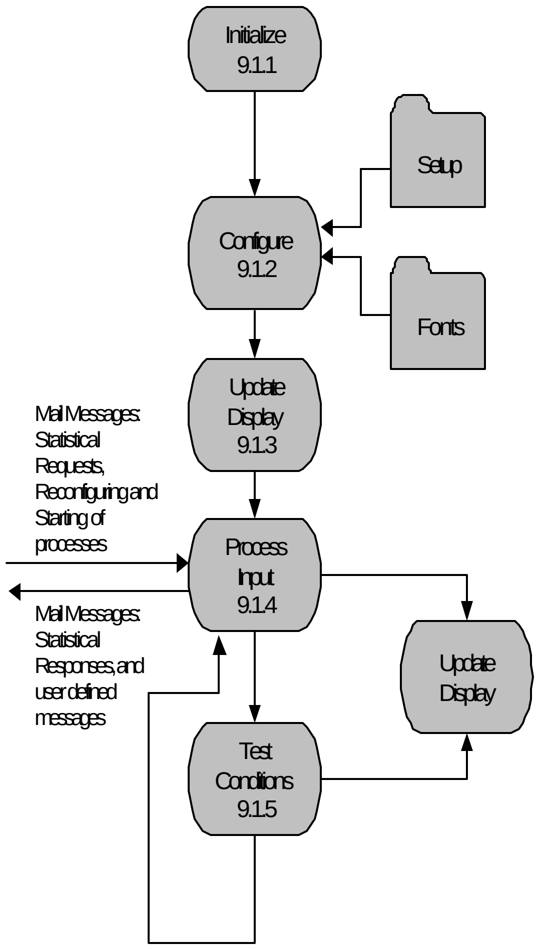
Initialize 9.1.1
Setup
Configure 9.1.2
Fonts
Update Display 9.1.3
Mail Messages: Statistical Requests, Reconfiguring and Starting of processes
Process Input 9.1.4
Mail Messages: Statistical Responses, and user defined messages
Update Display
Test Conditions 9.1.5

Figure 9-1. Data Flow for the Tool Manager

If no other file is passed as an argument during the initialize process, the *Configure* module reads the default **setup** file. If the **setup** file does not exist, then the default values set in the *Initialize* module are used and *TMGR* goes into the *Update Display* module. Otherwise, *TMGR* reads each line in the **setup** file and if the variable and the value are declared properly, the variable is set in the *Configure* module for displaying or processing. Any unknown variables, values or improper declarations are displayed on the screen.

The *Update Display* module is designed to draw the window based on the variables set in the *Initialize* and *Configure* modules. The *Update Display* module uses the Domain Graphic Primitives Routines (GPR) to display the window. The *clear_window, draw_title*, *draw_icons*, and *draw_border* procedures draw the display.

The operation of *TMGR* is driven by the *Process Input* and *Test Conditions* modules. The *Process Input* module repeatedly checks for keystrokes, cursor movements, and mailbox input. When a user command is entered, the *Process Input* module responds to the command. User commands are invoked by selecting icons and menu items. User commands can also be invoked by pressing defined *hot keys*. Hot keys are user-defined keys that perform the selection of icons or menu items, changing the look of the display or re-reading the **setup** file.

The *Process Input* module checks repeatedly for asynchronous mailbox input. When a mailbox input is received, *Process Input* performs the task involved for that mail message. The following are the tasks that *TMGR* responds to: statistical requests, reconfigure commands, or the starting of an icon record or menu item.

The *Test Conditions* module tests:

- Window movement
- Entering and leaving the window
- Timeout for animating icons
- Processes that were started and have exited
- Processes being started without *TMGR*
- Updating the display for the clock
- Determining whether the help window has been closed

These conditions can occur at any time. When they do occur, *TMGR* performs the necessary display updates or internal tables updates for each situation.

The *Initialize, Configure, Update Display, Process Input*, and *Test Conditions* modules communicate with each other through a vast array of global variables. Global variables were used because many routines are dependent on many parameters, and because the execution thread of *TMGR* can be extremely varied. For example, an array of process invocations are used for tracking what is invoked and the icon record associated with it. The **process_uids** array is used in the *Process Input* module for starting the icon record, the *Testing Conditions* module for checking whether the process has exited, and the *Update Display* module for re-drawing the display when the process has exited.

9.1.1 The Initialize Module

Purpose

Performs the initialization for *TMGR* function.

Input

The *Initialize* module reads the following optional arguments from the command line that initiates the program execution:

-h - This option displays argument options.

-p { (**X**-coordinate, **left**, **right**, **center**) (**Y**-coordinate, **top**, **bottom**, **center**) } **-** This option positions the Tool Manager window when the program starts.

-path - This option retrieves and stores data at a different directory.

-s { name of **s**etup file } **-** This option uses a different tool *TMGR* s**etup** file.

-v4 - This option prevents connection to a node switch at startup time.

Output

TMGR sets and creates an error log file named **/tmgr/trace/tmgr_crash**. If the file already exists, the size of the fileis checked. If the size of the file is greater than 4K, the file is deleted and a new file is created. Whenever *TMGR* crashes because of an error or the user forces the program to crash (by pressing **Ctrl Q**), the program writes the time and date when the crash occurred and the code that it was processing.

Processing

The *Initialization* module processing consists of a long series of steps, many of which are performed by invoking separate routines, executed sequentially from the main program module. Following is a summary of the initialization steps:

(1) Set the cleanup procedure whenever the program crashes unexpectedly. This procedure is invoked by the *cleanup_handler* routine.

(2) Find the name of the directory from which to retrieve and store data. This is done by calling the *get_etms_path* ETMS toolkit procedure.

(3) Set to their default values all variables used to display the *TMGR* window.

(4) Read the command line arguments and set a corresponding flag for each argument that exists. This procedure is invoked by the *get_arguments* routine.

(5) Connect to the communication program node switch if the user has not passed an argument in the command line.

If connection is successful, the "Connected to Node Switch" message is displayed on the screen. Otherwise, the "Node Switch Not Available! Could not connect to node switch. I will try again in one minute." warning message is displayed.

This procedure is involked by the *initialize_network_addressing* routine.

(6) Initialize the Domain Graphic Primitives Routines (GPR) and set the mode to direct.

(7) Create a window for displaying the menu items.

Error Conditions and Handling

Errors incurred during the *Initialize* module can be fatal or non-fatal.

Non-fatal Errors. Non-fatal errors cause the system to display an error message in the pad, but *TMGR* continues to execute. The following non-fatal errors may occur during the *Initialize* processing:

- The crash file is locked by another process. A different trace back file is then created.
- An improper argument for the **-p** command exists. (*get_arguments*)
- Could not connect to node switch. *TMGR* will try to connect in one minute. (*initialize_network_addressing*)

Fatal Errors. Fatal errors cause *TMGR* to terminate execution. The following fatal errors may occur during the *Initialize* processing:

- The name of the directory for retrieving and storing data is unknown or does not exist. (*get_etms_path*)
- An unknown argument is passed on the command line. (*get_arguments*)

The Cleanup Procedure. Before *TMGR* terminates its execution, the *cleanup_handler* routine invokes the cleanup procedure, which performs the following steps:

- Closes the node switch connection if there is a connection.
- Restores the prior color map and terminates graphics if the color map and graphics have been initialized.
- Deletes the temporary help file if the help window is displayed.

- Opens the crash file and writes the time, date, and the trace back information, if any, into the crash file.
- Restores original window size, font, and icon representation if they have been modified.
- Closes the menu window.
- Restores the normal input pad.

9.1.2 The Configure Module

Purpose

The *Configure* module reads the default **setup** file if no other file is passed as an argument during initialize procedure. If the default **setup** file does not exist, then the default values set in the *Initialize* module are used and *TMGR* goes into the *Update Display* module. Otherwise, *TMGR* reads each line in the **setup** file and if the variable and the value are declared properly, the variable is set in the *Configure* module for displaying or processing. Any unknown variables, values, or improper declarations are displayed on the screen.

Input

The *Configure* module is passed the name of the **setup** file that is to be read.

The *TMGR* function operates according to values that are assigned and defined in the **setup** file. The default **setup** file is **/tmgr/config/tmgr.default**. The **setup** file describes what is displayed on *TMGR* and what to execute when icons and menu items are selected.

The variables allow each *TMGR* executed to be configured for different needs or appearances. All variables and values can be upper or lower case. To assign a value to a variable, the variable must be followed by **:=** and then a value; for example, **variable:=value**. Comments are placed in the **setup** file by placing the symbol # as the first letter on the line. Comments make the file readable for inexperienced users.

NOTE: The variables listed as input in this section are fully described, beginning on page 9-2.

Setup File Variables. *TMGR* reads any of the following variables from the **setup** file when it is initially invoked:

animate_icons

animate_speed

children_hot_key

children_icons

digital_clock_on

digital_clock_position

digital_clock_time

digital_clock_24hour

digital_date

function_keys_on

help_file_region

help_icon_char

help_icon_on

icon_bg_color

icon_spacing

icon_text_color

invoke_icon_char

invoke_icon_on

invoke_region

lock_position

menu_bg_color

menu_font_title_path

menu_text_color

menu_title_bg_color

menu_title_text_color

orientation

quit_icon_char

quit_icon_on

read_setup_hot_key

title

title_bg_color

title_text_color

tmgr_bg_color

tmgr_icon_path

Icon Record Variables. The variables on the previous pages set *TMGR*'s colors and also the *Help, Invoke,* or *Quit* icons To further customize *TMGR*, an icon record is defined. Defining icon records allows users to start one or more different ETMS programs.

To begin a record, the line should contain the **icon_record:** variable. All variables beneath this line are associated with this one icon record. A maximum of 30 icons can be defined.

A special icon record named **default_icon_record:** can be declared to declare variables once rather than having copies throughout the **setup** file. This record uses the same variables as the icon record (except menu items). If a variable in an icon record is not declared but declared in the default, the icon record will use the value assigned in the default icon record. The end of an icon record is found when another icon record or default icon record is declared. Comments, the assignment symbol (**:=**), and upper and lower case rules still apply.

NOTE: The icon variables listed as input in this section are fully described beginning on page 9-6.

TMGR reads any of the following variables for icon records:

auto_close

auto_quit

help_file

hot_key

icon_char

icon_path

icon_word

menu_title

no_window

number_process_children

process_invoke

process_name

process_region

send_message

send_message_address

send_message_code

user_access

Menu Item Variables for Icons. There can be more than one **process_invoke** variable to each icon record by declaring menu items. Menu items allow more flexibility in starting the same process with different arguments. Menu items are defined inside an icon record. There is a maximum of 30 menu items allowed in an icon record.

To begin a menu item, the line should contain the **menu_item:** declaration. The menu item ends when another menu item, icon record, or default icon record is declared. A menu item can *inherit* values from the icon record or default icon record. The variables described on the following pages can inherit values.

NOTE: The menu item variables for icons listed as input in this section are fully described beginning on page 9-8.

All variables following the line containing **menu_time:** are associated with this one menu item. Comments, the assignment symbol (**:=**), and upper and lower case rules still apply. *TMGR* reads any of the following variables for menu items:

auto_close

auto_quit

heading

help_file

hot_key

no_window

process_invoke

process_region

send_message

send_message_address

send_message_code

Setup File Include Statement. To make the **setup** file more readable and modular, the **%include** statement is used. The parameter after the statement is another **setup** file that should be read. Inside this **include** file can be icon record(s), menu item(s), default icon record or variables. The name of the file can be a complete pathname or by itself. If the file is not a complete pathname, the location of the file is assumed to be in the */tmgr/config* directory.

For more information on the **include** file options, see page 9-10.

Output

The *Configure* module outputs any unknown variables, values, or improper declarations into the pad window. The output contains specific information to help the user solve the problem. For example, if the user did not use an assignment statement, the output will be as follows: **Missing := in setup file. Line #*n*** . The letter ***n*** represents the line number in the setup file that contains the error. This allows the user to easily find the line number and correct the problem.

Processing

The recursive main logic of the *Configure* module is shown in Figure 9-2.

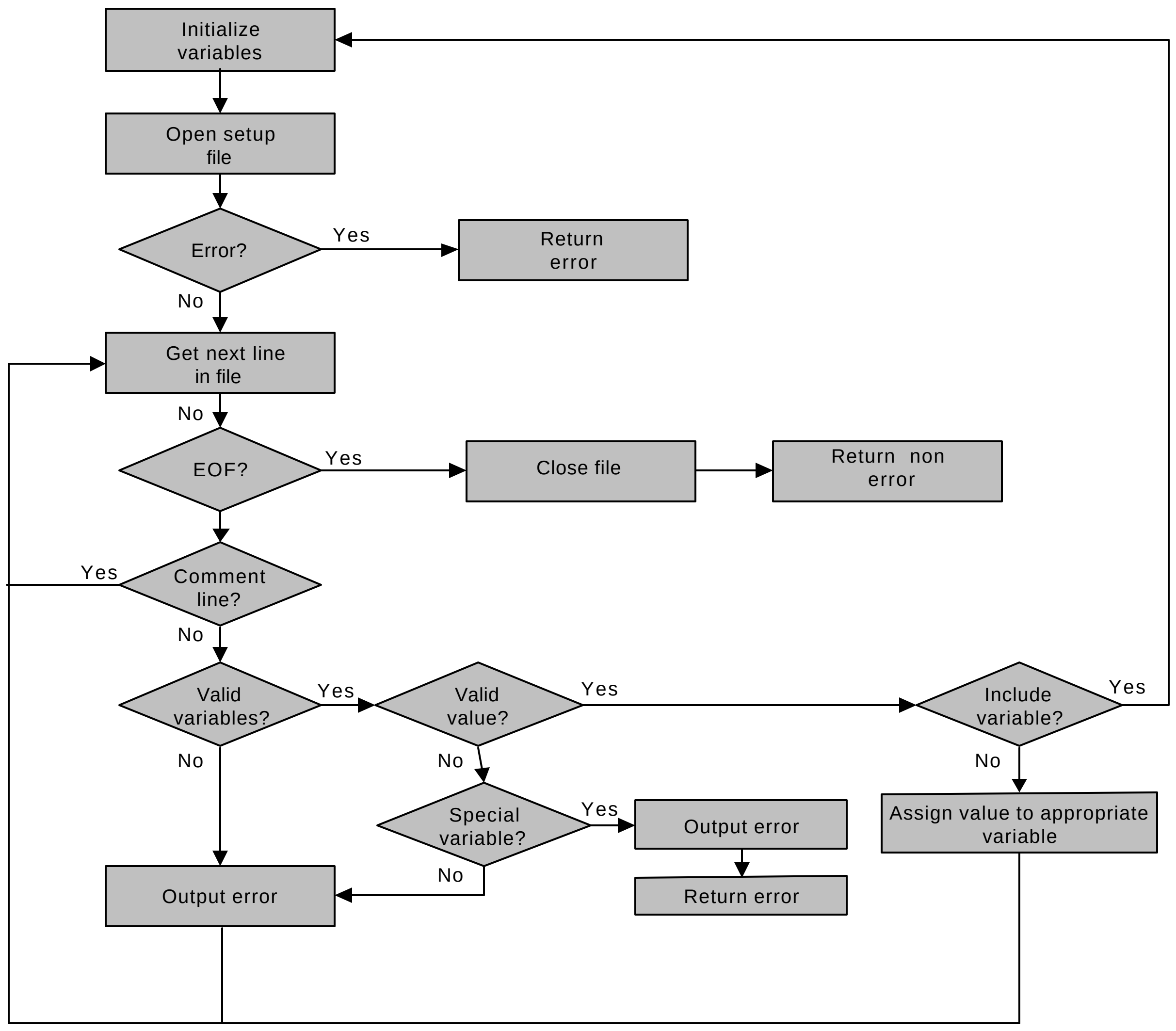

Figure 9-2. Main Logic for the Configure Module

The *Configure* module starts by reading the **setup** file after the *Initialize* module has completed the initializing processing and continues reading the file until a non-fatal error has occurred or the end of file (EOF) has been reached. If the variable is an appropriate variable with the correct value assigned to it, the value in the **setup** file is set to the internal variable in *TMGR*. After reading the **setup** file, the graphical locations for the icons are determined by the *setup_icons* procedure. This procedure sets the GPR data records for drawing the icons, fonts, and window size. The *setup_icons* procedure is invoked by *determine_window_size* procedure.

The *Initialize Variables* module initializes the global **default_record** variable if the setup file is not an include file. The *Configure* module originally reads a non include file from the main line. The procedure that performs the *Configure* is a recursive procedure (*read_setup_file*) for the original **setup** file and the **%include** variables. The *Initialize Variables* module also determines the current node that the program is running on and the user logged onto it. The current node and user code values are initialized for the **user_access** variable used in the **setup** file.

The *Special Variable* conditional module is designed to return an error when the variables **help_file** and **icon_path** have no value assigned on the current line and no corresponding data in the default record. This special condition checking allows the user to quickly fix the serious error. All other variables declared after this error will not be defined.

The *Assign Value to Appropriate Variable* module assigns the value on the line to the internal variable in *TMGR*. The internal variable could be the variable array **icon_record** if the variable is associated with an icon record or menu item, or the value assigned to the variables associated with the display of *TMGR*, such as **orientation**, **icon_spacing**, and so on.

Error Conditions and Handling

Errors incurred during the *Configure* module are all non-fatal. Non-fatal errors cause the system to display an error message in the pad, but *TMGR* continues to execute.

The following **non-fatal** errors may occur during the *Configure* processing:

- Unknown variable in setup file, which is detected by the *read_setup_file* procedure.
- Improper assignment statement for variable in setup file, which is detected by the *read_setup_file* procedure.
- Unknown or improper value assigned to variable, which is detected by the *read_setup_file* procedure.
- No value for variable, which is detected by the *read_setup_file* procedure.
- **Font**, **help**, or **include** file does not exist, which is detected by the *read_setup_file* procedure.

9.1.3 The Update Display Module

Purpose

The *Update Display* module is designed to draw the window based on the variables set in the *Initialize* and *Configure* modules. The *Update Display* module uses the Domain Graphic Primitives Routines (GPR) to display the window. The following procedures draw the display: *clear_window*, *draw_title*, *draw_icons*, and *draw_border*.

Input

title

digital_clock_on

Output

The *Update Display* module draws icons to the display.

Processing

The main logic of the *Update Display* module is shown in Figure 9-3.

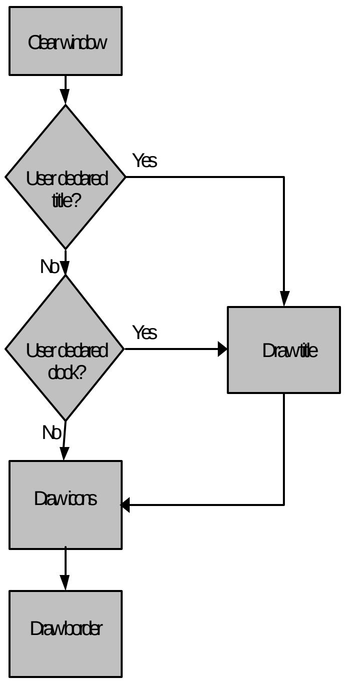

Figure 9-3. Main Logic for the Update Display Module

The *Update Display* module starts by using the *clear_window* procedure to clear the window to the user-defined color. If the user has defined data in the **title** or **digital_clock_on** variables, the *draw_title* procedure is then performed. The *draw_title* procedure calls *draw_clock* if the user has defined the **digital_clock_on**. If **digital_clock_on** is set to FALSE and **digital_date** is set to TRUE, the clock will not be drawn. The clock is only drawn if **digital_clock_on** is set to TRUE and then all other digital variables are appended to the current time.

The *Draw Icons* module draws the user-defined icons set in the *Initialize* and *Configure* modules. It draws the children processes started by the user and also the menu items if the icon is declared as a menu record. The *Draw Icons* module draws the icons by calling the *draw_icons* and *draw_icon* procedures. The *draw_icon* procedure uses the GPR routines for displaying the icons.

The *Draw Border* module draws a black border around the *TMGR* window (*draw_border*).

Error Conditions and Handling

The *Update Display* module is not designed to encounter any non-fatal or fatal errors. Any errors will be unexpected fatal errors that set off the *cleanup_handler* procedure. This procedure was discussed in Section 9.1.1.

9.1.4 The Process Input Module

Purpose

The operation of *TMGR* is driven by the *Process Input* and *Test Conditions* modules. The *Process Input* module repeatedly checks for keystrokes, cursor movements, and mailbox input. When a user command is entered, the *Process Input* module responds to the command. User commands are used to select icons and menu items. User commands can also be executed by pressing defined *hot keys*. Hot keys are user-defined keys that can select icons or menu items; change the look of the display, or re-reading the **setup** file.

The *Process Input* module checks repeatedly for asynchronous mailbox input. When a mailbox input is received, *Process Input* performs the task involved for that mail message. Tasks that *TMGR* responds to are statistical requests, reconfigure commands, or the starting of an icon record or menu item. The *Process Input* module invokes routines from *Update Display* to generate output on the screen.

Input

TMGR receives dynamic mail message data from other ETMS functions. *TMGR* processes the following mail messages:

- Statistical requests

- Reconfigure commands

- Start commands

Start commands allow a remote user to start any of the icon records or menu items controlled by *TMGR*. Receiving mail messages is discussed in Section 9.1.4.2.

The *Process Input* module optionally reads data from the following file type in response to user commands:

- **setup** file, which contains the variables, icon records, and menu items to display and monitor.

Output

TMGR sends dynamic mail message data to other ETMS functions. *TMGR* sends mail messages to any ETMS software described by the variable **send_message_address.** The content of the message is described by the variables **send_message** and **send_message_code**. If all these variables are declared and properly assigned, *TMGR* passes the information to the communication software running on the computer.

TMGR optionally writes data to the following files in response to user commands:

- A temporary help file **-** *TMGR* creates a file in the **/tmp directory** when the user requests a help file. This file is displayed in a window until *TMGR* quits or the user closes the window with the correct Display Manager commands. The file is deleted when the window is closed.
- A crash file **-** *TMGR* creates an error log file named ***/tmgr/trace/tmgr_crash***. If the file already exists, the size of the file is checked. If the size of the file is greater than 4K, the file is deleted and a new file is created. Whenever *TMGR* crashes because of an error or the user forces the program to crash (pressing **Ctrl Q**), the program writes the time and date when the crash occurred and the current code that it was processing.

The *Process Input* module sends display parameters to the *Update Display* routines. These parameters include the following:

- Current window size
- Color map
- Cursor position
- Event type

Processing

The main logic of the *Process Input* module is shown in Figure 9-4.

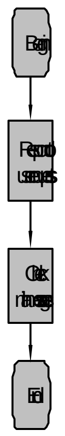

Figure 9-4. Main Logic for the Process Input Module

The *Process Input* module starts after the *Update Display* module has completed drawing the display. The *Process Input* module determines whether the program should exit after the user selects the *quit* icon or presses the **Q** key on the keyboard.

The *Respond to User Requests* routine checks for any user input. The user enters requests in two ways: typing single keystrokes or selecting an icon or menu item with the left mouse button. If any input is present, the *Respond to User Requests* routine obtains whatever information it needs from the user and invokes a routine to execute the request. The *Respond to User Requests* routine is described below.

The *Check Mail Messages* routine checks for messages from the communication process. The *Check Mail Messages* routine responds to statistical requests, and reconfigure and start commands. The *Check Mail Messages* routine is described in Section 9.1.4.2.

9.1.4.1 The Respond to User Requests Routine

The *Respond to User Requests* routine accepts two general classes of input from the user: keyboard commands and mouse selections on icons or menu items. Keyboard commands are single keystrokes. Mouse selections are done by releasing the mouse button when the cursor is on an icon or menu item.

When the cursor enters an icon or menu item, the item changes color. The item changes into reverse video. The background color changes into the text color and the text color changes into the background color. This visual clue allows the user to see that the item is *selectable*.

The *Respond to User Requests* routine first checks whether a keystroke has been entered. There are the following five possible interpretations of the entered keystroke, in the order they are checked:

- Children icons - on or off
- Tab or Shift-Tab event
- User-defined hot key
- Function key
- Quit key

Toggling the children on or off on the display is defined as a **control-k** (default) or is user-defined in the **setup** file. Displaying the children will show small numbered icons at the lower right of the icon that it started from.

Pressing the **Tab** key places the cursor at the first icon on the display if the cursor is not on an icon, or it advances the cursor to the lower right of the next icon on the display. The **Shift Tab** keystroke places the cursor on the last icon if the cursor is not on an icon, or it places the cursor to the upper left of the icon that it is currently on. If the menu items are displayed, the **Tab** and **Shift Tab** keys will work on the menu items rather than the icons. The **Tab** and **Shift Tab** perform a wrap-around when the cursor reaches the first or last items.

The user-defined hot key is a keystroke defined in the **setup** file to perform the startup of an icon. The user-defined key can be for an icon record or menu item.

The user can also start an icon record by pressing a function key to the corresponding icon record. The function keys **F1** to **F9** will start the icons numbered 1 through 9. The function key **F0** will start the Invoke icon (if declared in the **setup** file). Selecting **F0** will display the prompt for invoking a shell command. The function keys will work only if the variable **function_keys_on** is set to TRUE.

Pressing the upper or lower case letter **Q** will quit the program if the above events have failed.

If a keystroke has not been entered, the *Respond to User Requests* routine checks for the left mouse button release. If the cursor is on an icon or menu item, the appropriate process for the item is started. If the user has reached the maximum allowed processes to start (**number_process_children**), *TMGR pops* the first started process window to the front. If the left button is released on a child icon, the process window represented by that icon is popped to the front of all windows. The release of the left button also pops the process window to the front if the process window was displayed as an icon.

The logic for processing a user request is shown in Figure 9-5.

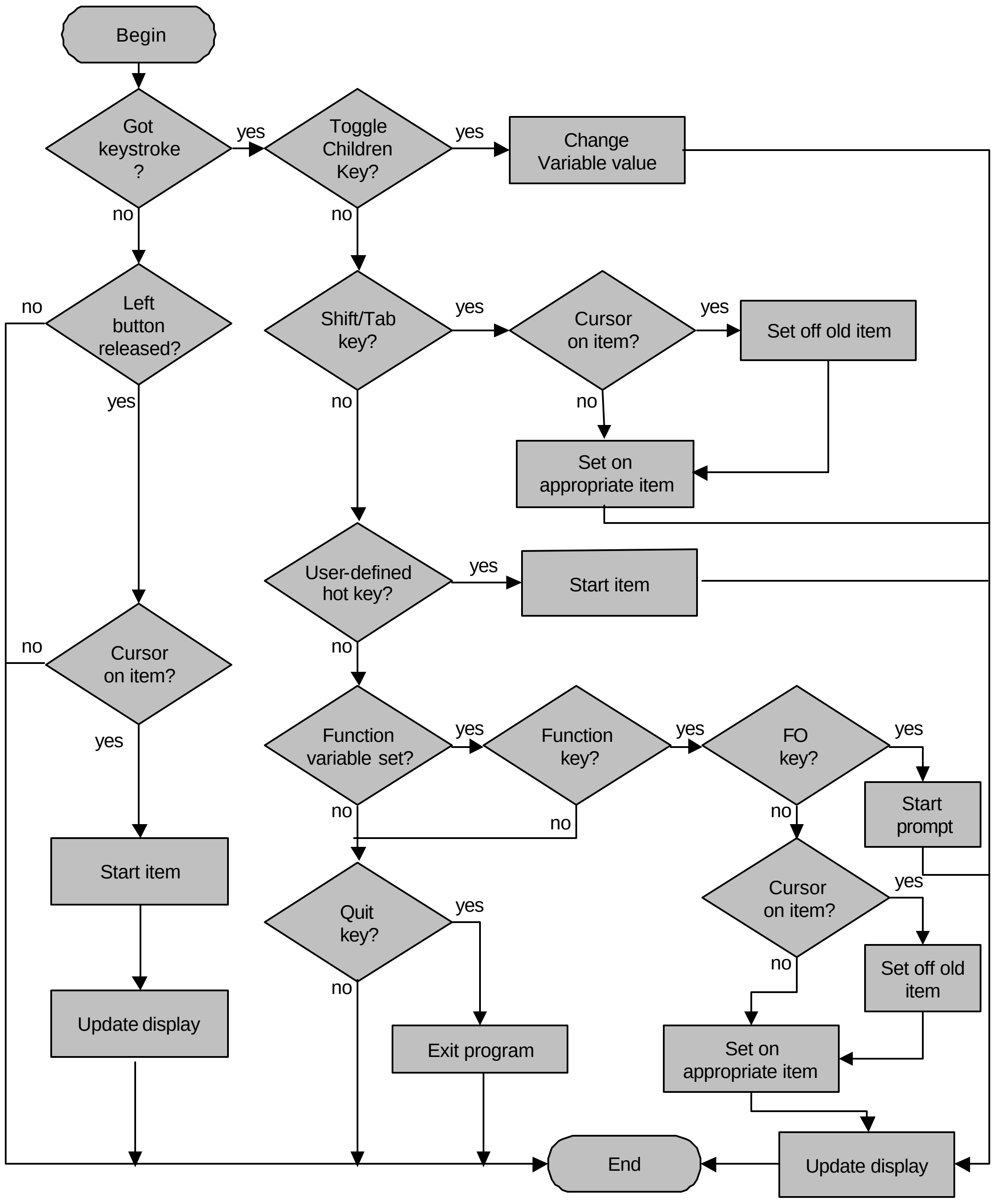

Figure 9-5. Logic for the Respond to User Request Routine

The *Cursor On Item* routine determines whether the cursor is currently on an icon, children icon, or menu item. This is done by checking the cursor **x** and **y** position to the location and width and height of the item it is examining. The *Cursor On Item* routine returns the index number corresponding to the icon, children icon, or menu item that the cursor is on. Otherwise the routine returns a **-1** to designate that the cursor is not on an item.

The *Set Off Old Item* accepts the value returned from *Cursor On Item* and changes the **icon_on** variable in the **icons** record variable to FALSE. The *Set On Appropriate Value* accepts the value returned from *Cursor On Item* and changes the **icon_on** variable in the **icons** record variable to TRUE. The *Update Display* module then draws the screen with the correct settings.

The *Start Item* routine accepts the index to the icon record or menu item that is being started. The index points to the global variable **icons** that are being started. The variable contains all information that users defined for the icon, including the menu items. A boolean variable **found_menu_hot_key** determines whether the index is for a regular icon record or a menu item inside the icon record. If the variable **found_menu_hot_key** is set to TRUE, the index is pointing to a menu item, otherwise the routine uses the data not declared as a menu item. The *Start Item* routine starts the information declared in the icon or menu item. The data can be ETMS software; ETMS mail messages, Display Manager Commands, or Shell Commands.

The *Start Prompt* routine is started when the user presses the **F0** key, if it is defined, or selects the *invoke* icon with the left mouse button. The routine begins by displaying a rectangular box. In the far right of the box **Invoke:** appears. The purpose of the *invoke* icon is to allow entry of Apollo Domain shell commands without creating a window. For example, if the Traffic Manager specialist wanted to do a listing of a directory, that individual would select the icon and type **ld /traffic** and press the **Return** key. A window will appear with the output generated from the command. The user can press the **Esc** key if a shell command is not needed.

The *Update Display* module is discussed in Section 9.1.3.

9.1.4.2 The Check Mail Messages Routine

The *Check Mail Messages* routine is executed repeatedly as part of the main loop of the *Process Input* module of *TMGR*. On each loop, the *Check Mail Messages* routine determines whether any new mail input is available. If so, this routine gets the input data and determines what to do with the data.

The logic of the *Check Mail Messages* routine is shown in Figure 9-6. The global flags **connect_to_node_switch** and **connection_handle** are each checked to see whether the user wants to connect to the communication software and to determine whether the connection is valid. If the **connect_to_node_switch** flag is set, the routine *check_port* is invoked to test for data and process the data, if any. The three tests that *Check Mail Messages* performs are statistical requests, and reconfigure and start commands.

An ETMS program can send statistical requests to *TMGR*. *TMGR* receives the requests and returns a message containing the icon records and menu items defined in the **setup** file. The following is an example of a statistical reply:

```
 ***TMGR {Version} ***

Current time: {Current Time}

Start Time: {Start Time}
Current configuration file: {Name of setup file}

              Children                                  Auto
Process       max  run    Hot Key          Quit         Window   Invocation

{Process Name} {#} {#}  {Key Defined} {True/False} {True/False}  {Invoke}
{Process Name}
  1) {Heading}    {#} {#}  {Key Defined} {True/False} {True/False}  {Invoke)
  2) {Heading}    {#} {#}  {Key Defined} {True/False} {True/False}  {Invoke)
{Process Name} {#} {#}  {Key Defined} {True/False} {True/False}  {Invoke}
```

Reconfigure commands received through the communication process tell *TMGR* to read a **setup** file. The **setup** file can be passed as the next argument after the reconfigure command or no argument at all. If the **setup** file is not given, *TMGR* uses the current configuration as the **setup** file. *TMGR* uses the *Configure* module as described in Section 9.1.2 after the reconfigure command is received.

The start command is used to start a process as though the user selected it. This command allows a remote user to start a process managed by *TMGR*. The following is the format for starting an icon record: *start {Process Name}*. Remote users can also start menu items: *start {Process Name} {#}*. The number sign, *#* , designates which menu item in the record to start. The number is shown in the statistical reply from *TMGR*. This is shown in the preceding example.

The statistics, reconfigure, and start requests can all be done through the ETMS function *Netmail*.

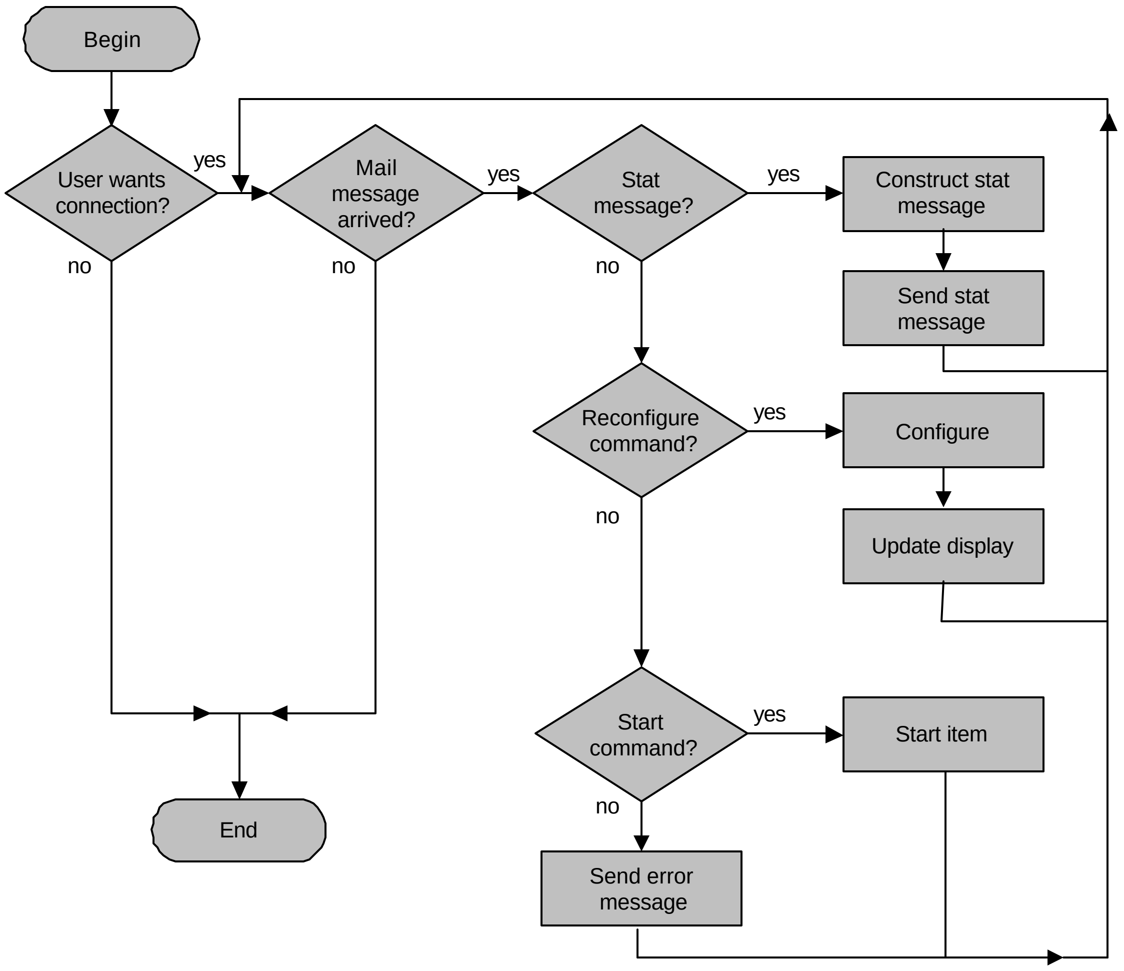

Figure 9-6. Logic for the Check Mail Messages Routine

Error Conditions and Handling

Errors incurred during the *Check Mail Messages* routine are all non-fatal. Non-fatal errors cause the system to display an error message in the pad, but *TMGR* continues to execute.

The following **non-fatal** errors may occur during the *Check Mail Messages* processing:

- The **process_invoke** variable can be declared improperly and the icon record cannot be started. Apollo Domain system displays an error message in Alarm Box pad.

- **Setup** file does not exist when passed in the reconfigure command (*check_port*). *TMGR* sends an error message to the user sending the command, a window appears for two seconds to the user describing the error, and the current **setup** file is used for the reconfigure command.
- Unknown process name is given in the start command (*check_port*). An error message is returned to the sender and the Start command is not performed.
- Unknown menu item is given in the start command (*check_port*). An error message is returned to the sender and the Start command is not performed.
- Unknown command is sent to *TMGR*. An error message is returned to the sender.

9.1.5 The Test Conditions Module

Purpose

The *Test Conditions* module tests the following situations:

- Window movement
- Entering and leaving of window
- Time-out for animating icons
- Processes that were started and have exited
- Processes being started without *TMGR*
- Updating the display for the clock and to determine if the *help* window has been closed

These conditions can occur at any time. When they do occur, *TMGR* performs the necessary display updates or internal tables updates for each situation.

Input

The *Test Conditions* routines receive the following data from the *Initialize*, *Configure*, and *Process Input* modules:

- Icon records defined in the setup file. The variable is named **icons**.
- Processes started by *TMGR*. The variable array **process_uids** tells the child number and related icon that it started from.
- The stream identifier created when user requests a help window for an item. The variable is named **help_file_stream**.

Output

The *Test Conditions* routines update the display when processes are started or exited.

Processing

The main logic of the *Test Conditions* module is shown in Figure 12-7.

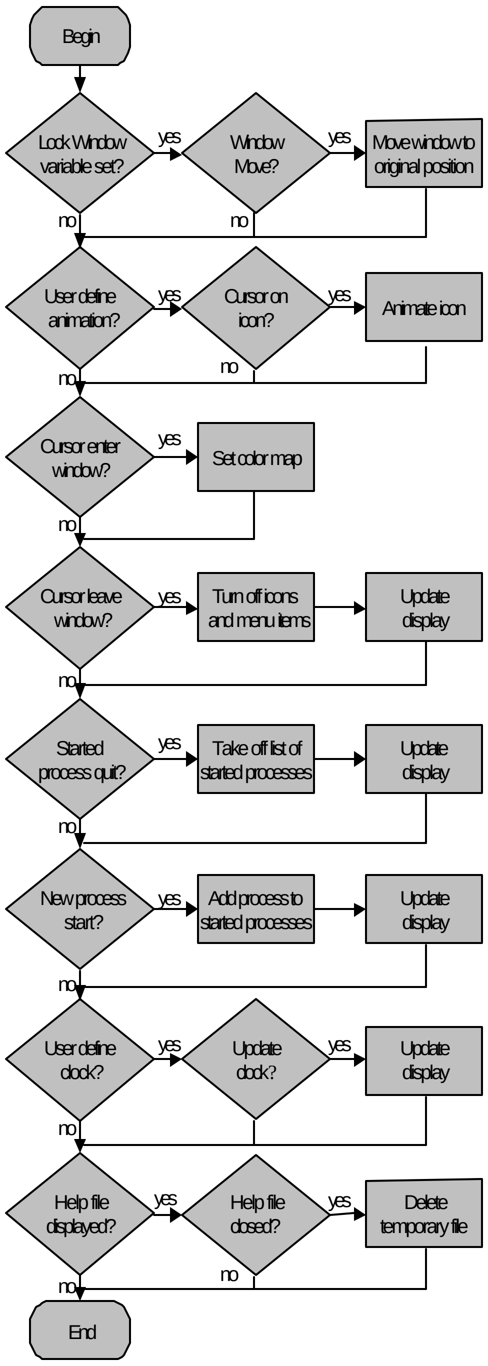

Figure 9-7. Main Logic for the Test Conditions Module

The *Test Conditions* module starts after the *Process Inputs* module has completed testing user and mail inputs. After the *Test Conditions* module finishes all the condition testing, the processing returns to the *Process Inputs* module.

The *Cursor Enter Window* routine checks to see if the cursor has entered the *TMGR* window. When this event occurs, *TMGR* restores the color map section that it works in. This event is important because other applications on the computer can modify the color map. When the user moves the cursor in *TMGR's* window, the icons and menu items will be visible with the appropriate colors.

The *Cursor Leave Window* routine checks to see if the cursor has left the *TMGR* window. When this event occurs, all icons or menu items that were turned to reverse video are changed back to their original colors. All items should not be highlighted when the cursor is not physically on them. The *Update Display* module was described in Section 9.1.3.

The *Started Process Quit* routine determines if any process managed by *TMGR* has exited. If the process has exited, *TMGR* draws the display without the appropriate child icon beneath (right) the parent (assuming the children icons are displayed). The routine will not manage the child any more by taking it off the variable array **process_uids**. The *Started Process Quit* determines if a process has exited by listing the contents of a directory. The name of the directory is: */sys/node_data/proc_dir.* Located inside the directory are the names of processes running on the computer. If a process being tracked in the variable **process_uids** is not in the directory, the process has exited.

The *New Process Start* routine determines if a process defined in *TMGR* has been started without using *TMGR.* If the process is started without *TMGR*, the process will then be managed by displaying the child icon beneath (right) the parent icon (assuming the children icons are to be displayed). The process is also added to the variable array **process_uids**. The *New Process Start* routine determines if the process has been started by listing the contents of a directory. The name of the directory is: */sys/node_data/proc_dir.* Located inside the directory are the names of processes running on the computer. If a name of a process in the directory corresponds to a process defined in the **setup** file and the process is not already being managed, then add this found process to the variable **process_uids**.

The *Help File Displayed* routine determines if the help window is displayed. If the user displays a help window, the window can be closed by a Display Command keystroke. When the window is closed, *TMGR* deletes the file corresponding to the window. The location of the file is in the directory: */tmp*. The name of the file is **tmgr_help_{Time Stamp}**.

Error Conditions and Handling

The *Test Conditions* module is not designed to encounter any non-fatal or fatal errors. Any errors will be unexpected fatal errors that set off the *cleanup_handler* procedure. This procedure is discussed in Section 9.1.1.

9.1.6 TMGR Source Code Organization

This section describes the source code used in building the executable version of *TMGR*. The source code resides in Pascal files. Each file contains one or more functional units called a *routine*. A routine is implemented as either a Pascal function or procedure.

Before *Tool Manager* can be executed, the following commands must be issued in order to compile the appropriate files:

- **pas main_tmgr.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_utils.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_prompt.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_read.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_config.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_net.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_pop.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_window.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_invoke.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_error.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_cleanup.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_arguments.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_init.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_directory.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_draw.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_setup.pas -nopt -dba -comchk -subchk -cpu any**
- **pas tmgr_refresh.pas -nopt -dba -comchk -subchk -cpu any**

Once the files have been compiled, the following command must be issued in order to bind the files together into one executable program:

```
bind
          main_tmgr.bin
          tmgr_utils.bin
          tmgr_prompt.bin
          tmgr_read.bin
          tmgr_config.bin
          tmgr_net.bin
          tmgr_pop.bin
          tmgr_invoke.bin
          tmgr_window.bin
          tmgr_error.bin
          tmgr_cleanup.bin
          tmgr_arguments.bin
          tmgr_init.bin
          tmgr_directory.bin
          tmgr_draw.bin
          tmgr_setup.bin
          tmgr_refresh.bin
          get_etms_path
          toolkit
-b tmgr
```

Two files, **get_etms_path** and **toolkit,** are to be bound with ETMS software. These files contain the proper routines to do the data directory determination and interprocess communication calls.

9.1.7 *TMGR* Data Structure Tables

The Pascal record definitions of all data files are in **tmgr.ins.pas**. See Tables 9-1 through 9-5.

Table 9-1. TMGR icon_record Record

icon_record			
Library Name:		**Element Name:** tmgr.ins.pas	
Purpose: To store and display icon data after reading the setup file for TMGR.			
Variable Fields	**Type**	**Description**	**Variable Represented In Setup File**
icon_data	graphics_icon_rec	Store related graphics data for displaying icon.	icon_word
child_data	children_icon_rec	Store graphical	Not Applicable

Table 9-1. TMGR icon_record Record			
		and process names for started children.	
active_nums	array[1..max_children] of integer	List of available children numbers to use.	Not Applicable
max_children	integer	Maximum number of children to invoke for this icon.	number_process_children

Table 9-1. TMGR icon_record Record (continued)

icon_record			
Library Name:		**Element Name:** tmgr.ins.pas	
Purpose: To store and display icon data after reading the setup file for TMGR.			
Variable Fields	**Type**	**Description**	**Variable Represented In Setup File**
setup_font_char	name_$pname_t	Font characters to display on icon. Multiple of characters are used because of animation.	icon_char
font_char_length	integer	Number of font characters in the variable **setup_font_char**.	Not Applicable
current_font_index	integer	Font character currently displayed on icon. Multiple font characters are used because of animation. The number points to the font used in **setup_font_char**.	Not Applicable
setup_font_path	name_$pname_t	Name of font to use when displaying icon.	icon_path
setup_name	name_$pname_t	Process name to	process_name

		be given for this icon when the icon is started.	
setup_region	array[1..max_children] of region_type	Regions to use for each process started by icon.	process_region
setup_invoke	name_$pname_t	Invocation string set in setup file. This variable is used to start the process.	process_invoke
send_message	name_$pname_t	The message to send to a process if the icon was set for mailing messages.	send_message
send_message_ address	name_$pname_t	The address of the process to send a message when the icon is started.	send_message_address

Table 9-1. TMGR icon_record Record (continued)

icon_record			
Library Name:		**Element Name:** tmgr.ins.pas	
Purpose: To store and display icon data after reading the setup file for TMGR.			
Variable Fields	**Type**	**Description**	**Variable Represented In Setup File**
help_file	name_$pname_t	The name of the file to display when the user wants help information on the icon.	help_file
menu_pointer	menu_pointer	The list of menu records if the icon record contain menu items.	Menu_Item
send_message_ code	integer	The code to send to a process if the icon was set for mailing messages.	send_message_code
input_window	boolean	A true or false	invoke_icon_on

		value indicating if the icon invokes the prompt window for doing a shell command.	
auto_quit	boolean	A true or false value indicating if TMGR wll exit after the icon is selected.	auto_quit
auto_close	boolean	A true or false value indicating if the icon will automatically close the process window after the process has exited.	auto_close
no_window	boolean	A true or false value indicating if a window should be created for the process after the icon is selected.	no_window
hot_key	char	The key represented to start the icon.	hot_key

Table 9-2. TMGR menu_record Record

menu_record			
Library Name:		**Element Name:** tmgr.ins.pas	
Purpose : To store menu data after reading the setup file for TMGR.			
Variable Fields	**Type**	**Description**	**Variable Represented In Setup File**
icon_data	graphics_icon_rec	Store related graphics data for displaying icon.	icon_word
process_invoke	name_$pname_t	Invocation string set in setup file. This variable is used to start the process.	process_invoke
send_message	name_$pname_t	The message to send to a process if the icon was set for mailing messages.	send_message
send_message_ address	name_$pname_t	The address of the process to send a message when the icon is started.	send_message_addr ess
process_region	array[1..max_childr en] of process_region	Regions to use for each process started by icon.	process_region
send_message_ code	integer	The code to send to a process if the icon was set for mailing messages.	send_message_code
auto_quit	boolean	A true or false value indicating if TMGR wll exit after the icon is selected.	auto_quit
auto_close	boolean	A true or false value indicating if the icon will automatically close the window after the process has exited.	auto_close
no_window	boolean	A true or false value indicating if a window should be created for the process after the icon is selected.	no_window
next	menu_ptr	A pointer to the	Not Applicable

		next menu record.	

Table 9-3. TMGR process_uid_record Record

process_uid_record			
Library Name:		**Element Name:** tmgr.ins.pas	
Purpose : To store information about process's managed by TMGR			
Variable Fields	**Type**	**Description**	
process_name	name_$pname_t	Name of actual process name running on Operating System	
menu_heading	name_$pname_t	Internal variable to flag this process with a particular menu item if the process is related to a menu item.	
menu_length	integer	Length of the **menu_heading** variable.	
menu_process	boolean	A true or false value designating if this process is a menu item.	
process_length	integer	Length of the **process_name** variable	
parent_window	integer	Index that points to icon_record that started process	
child_window	integer	Index that points to child_data of child_icon_rec	

Table 9-4. TMGR graphics_icon_record Record

graphics_icon_record			
Library Name:		**Element Name:** tmgr.ins.pas	
Purpose : Record information for displaying icons.			
Variable Fields	**Type**	**Description**	
word	array[1..3] of name_$pname_t	A variable containing characters designating a particular picture or words that user defined in setup file.	
length	array[1..3] of integer	Lengths of the **word** variable.	
window	gpr_$window_t	X, y, width and height of the icon displayed.	
levels	integer	The the number of levels described for the **word** and **length** variables.	
big_word	integer	An index as to the longest word in the **word** variable.	
icon_on	boolean	A true or false value designating if the icon is highlighted.	
icon_font	integer	The font to display the icon.	
word_icon	boolean	A true or false value indicating if the icon is a word that user defined in the setup file.	

Table 9-5. TMGR children_icon_rec Record

children_icon_record			
Library Name:		**Element Name:** tmgr.ins.pas	
Purpose : Record information for displaying children icons.			
Variable Fields	**Type**	**Description**	
child_window	array[1..max_children] of gpr_$window_t	X, y, width and height of children icon displayed.	
child_name	array[1..max_children] of name_$pname_t	Name of actual process running on the Operating System.	
child_length	array[1..max_children] of integer	Length of the **child_name** variables.	
child_icon_on	array[1..max_children] of boolean	A true or false value indicating if the child icon is displayed.	
number_childs	integer	Number of children icons started by icon record.	
child_font	integer	Font to display the child icon number.	